Red Hat OpenShift Service auf AWS mit FSxN

NetApp container solutions

NetApp
January 21, 2026

Inhalt

Red Hat OpenShift Service auf AWS mit FSxN

Red Hat OpenShift Service auf AWS mit NetApp ONTAP

Überblick

In diesem Abschnitt zeigen wir, wie FSx für ONTAP als persistente Speicherschicht für auf ROSA ausgeführte Anwendungen genutzt wird. Es wird die Installation des NetApp Trident CSI-Treibers auf einem ROSA-Cluster, die Bereitstellung eines FSx für ONTAP -Dateisystems und die Bereitstellung einer Beispielanwendung mit Status gezeigt. Außerdem werden Strategien zum Sichern und Wiederherstellen Ihrer Anwendungsdaten gezeigt. Mit dieser integrierten Lösung können Sie ein gemeinsam genutztes Speicherframework einrichten, das sich mühelos über AZs skalieren lässt und die Prozesse der Skalierung, des Schutzes und der Wiederherstellung Ihrer Daten mithilfe des Trident CSI-Treibers vereinfacht.

Voraussetzungen

- "AWS-Konto"
- "Ein Red Hat-Konto"
- IAM-Benutzer"mit entsprechenden Berechtigungen" zum Erstellen und Zugreifen auf den ROSA-Cluster
- "AWS CLI"
- "ROSA CLI"
- "OpenShift-Befehlszeilenschnittstelle"(oc)
- Helm 3"Dokumentation"
- "Ein HCP ROSA-Cluster"
- "Zugriff auf die Red Hat OpenShift-Webkonsole"

Dieses Diagramm zeigt den in mehreren AZs bereitgestellten ROSA-Cluster. Die Masterknoten und Infrastrukturknoten des ROSA-Clusters befinden sich im VPC von Red Hat, während sich die Worker-Knoten in einem VPC im Konto des Kunden befinden. Wir erstellen ein FSx für ONTAP Dateisystem innerhalb derselben VPC und installieren den Trident -Treiber im ROSA-Cluster, sodass alle Subnetze dieser VPC eine Verbindung zum Dateisystem herstellen können.

Ersteinrichtung

1. Bereitstellung von FSx für NetApp ONTAP

Erstellen Sie ein Multi-AZ FSx für NetApp ONTAP in derselben VPC wie der ROSA-Cluster. Hierzu gibt es mehrere Möglichkeiten. Die Details zur Erstellung von FSxN mit einem CloudFormation Stack werden bereitgestellt

a.Klonen Sie das GitHub-Repository

```
$ git clone https://github.com/aws-samples/rosa-fsx-netapp-ontap.git
```

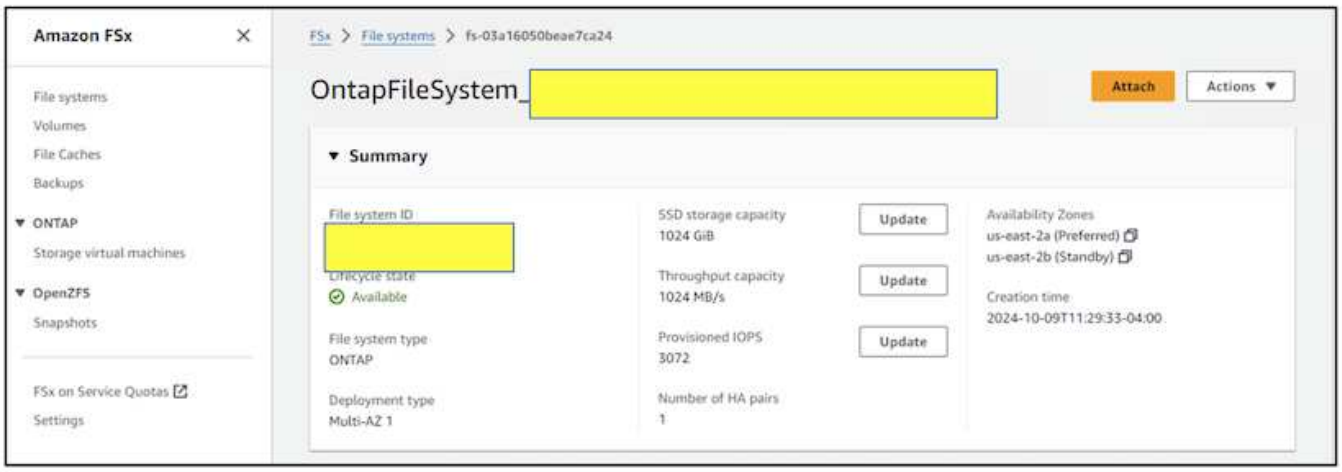
b. Führen Sie den CloudFormation-Stack aus. Führen Sie den folgenden Befehl aus, indem Sie die Parameterwerte durch Ihre eigenen Werte ersetzen:

```
$ cd rosa-fsx-netapp-ontap/fsx
```

```
$ aws cloudformation create-stack \
  --stack-name ROSA-FSXONTAP \
  --template-body file://./FSxONTAP.yaml \
  --region <region-name> \
  --parameters \
  ParameterKey=Subnet1ID,ParameterValue=[subnet1_ID] \
  ParameterKey=Subnet2ID,ParameterValue=[subnet2_ID] \
  ParameterKey=myVpc,ParameterValue=[VPC_ID] \
ParameterKey=FSxONTAPRouteTable,ParameterValue=[routetable1_ID,routetable2
_ID] \
  ParameterKey=FileSystemName,ParameterValue=ROSA-myFSxONTAP \
  ParameterKey=ThroughputCapacity,ParameterValue=1024 \
  ParameterKey=FSxAllowedCIDR,ParameterValue=[your_allowed_CIDR] \
  ParameterKey=FsxAdminPassword,ParameterValue=[Define Admin password] \
  ParameterKey=SvmAdminPassword,ParameterValue=[Define SVM password] \
  --capabilities CAPABILITY_NAMED_IAM
```

Wobei: Regionsname: identisch mit der Region, in der der ROSA-Cluster bereitgestellt wird. Subnet1_ID: ID des bevorzugten Subnetzes für FSxN. Subnet2_ID: ID des Standby-Subnetzes für FSxN. VPC_ID: ID der VPC, in der der ROSA-Cluster bereitgestellt wird. Routetable1_ID, Routetable2_ID: IDs der Routentabellen, die mit den oben ausgewählten Subnetzen verknüpft sind. Your_allowed_CIDR: zulässiger CIDR-Bereich für die Eingangsregeln der FSx for ONTAP Sicherheitsgruppen zur Zugriffskontrolle. Sie können 0.0.0.0/0 oder ein beliebiges geeignetes CIDR verwenden, um dem gesamten Datenverkehr den Zugriff auf die spezifischen Ports von FSx für ONTAP zu ermöglichen. Administratorkennwort definieren: Ein Kennwort zum Anmelden bei FSxN. SVM-Kennwort definieren: Ein Kennwort zum Anmelden bei SVM, das erstellt wird.

Überprüfen Sie, ob Ihr Dateisystem und Ihre virtuelle Speichermaschine (SVM) mithilfe der Amazon FSx Konsole erstellt wurden, wie unten gezeigt:



2. Installieren und konfigurieren Sie den Trident CSI-Treiber für den ROSA-Cluster

b.Trident Trident

ROSA-Cluster-Workerknoten sind mit NFS-Tools vorkonfiguriert, die Ihnen die Verwendung von NAS-Protokollen für die Speicherbereitstellung und den Speicherzugriff ermöglichen.

Wenn Sie stattdessen iSCSI verwenden möchten, müssen Sie die Worker-Knoten für iSCSI vorbereiten. Ab der Trident Version 25.02 können Sie die Worker-Knoten des ROSA-Clusters (oder eines beliebigen OpenShift-Clusters) problemlos für die Durchführung von iSCSI-Vorgängen auf FSxN-Speicher vorbereiten. Es gibt zwei einfache Möglichkeiten, Trident 25.02 (oder höher) zu installieren, das die Vorbereitung des Worker-Knotens für iSCSI automatisiert. 1. Verwenden Sie das Node-Prep-Flag über die Befehlszeile mit dem Tool Tridentctl. 2. Verwenden Sie den von Red Hat zertifizierten Trident Operator vom Operator-Hub und passen Sie ihn an. 3. Helm verwenden.

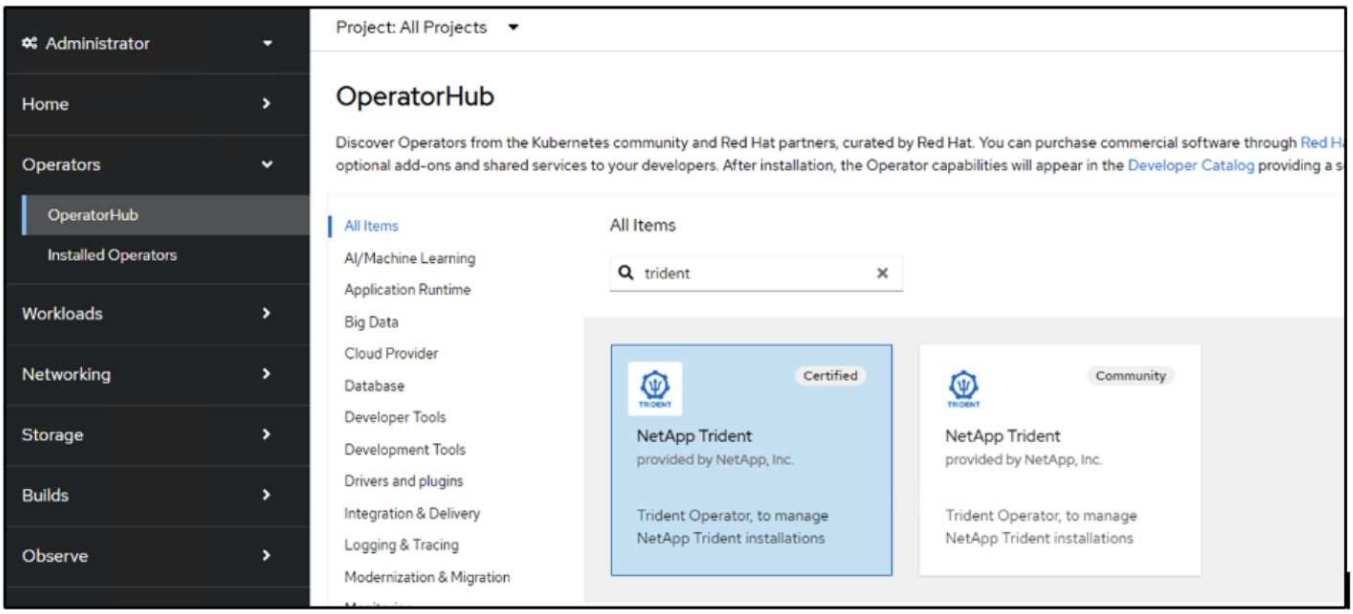
Wenn Sie eine der oben genannten Methoden verwenden, ohne die Knotenvorbereitung zu aktivieren, können Sie für die Bereitstellung von Speicher auf FSxN nur NAS-Protokolle verwenden.

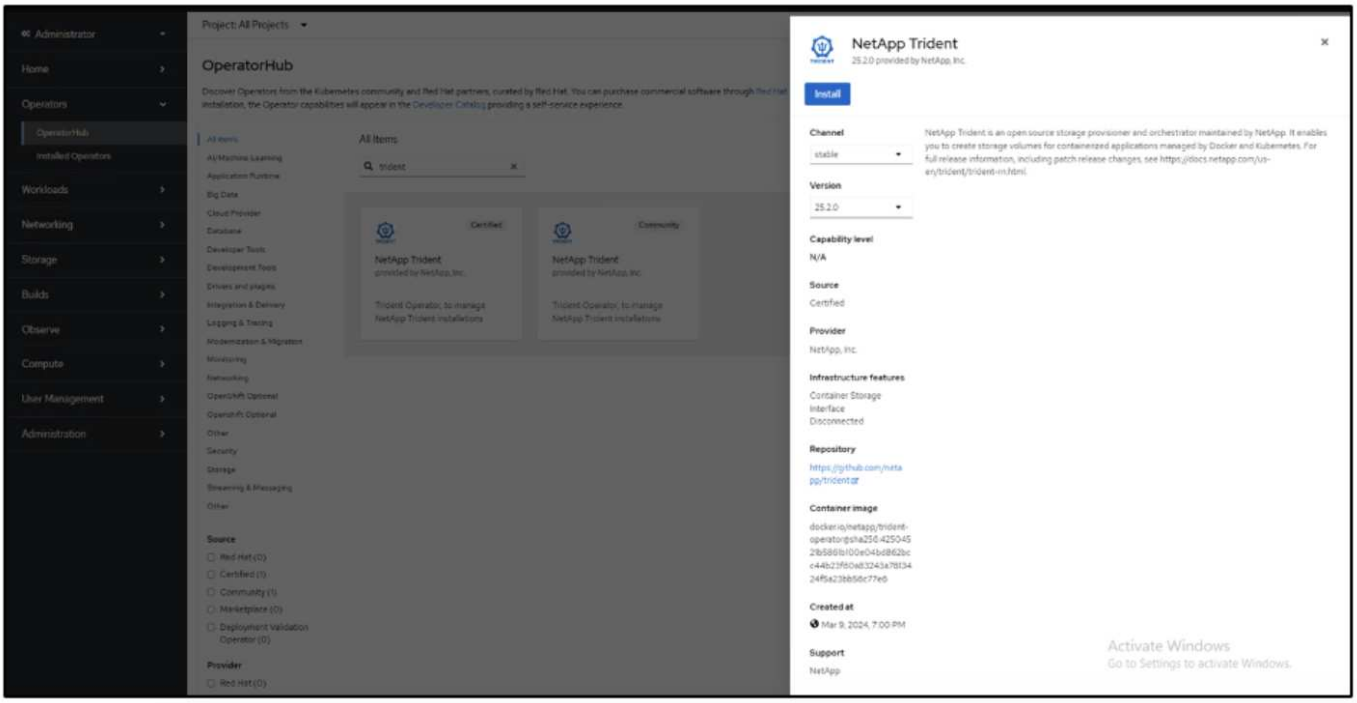
Methode 1: Verwenden Sie das Tool tridentctl

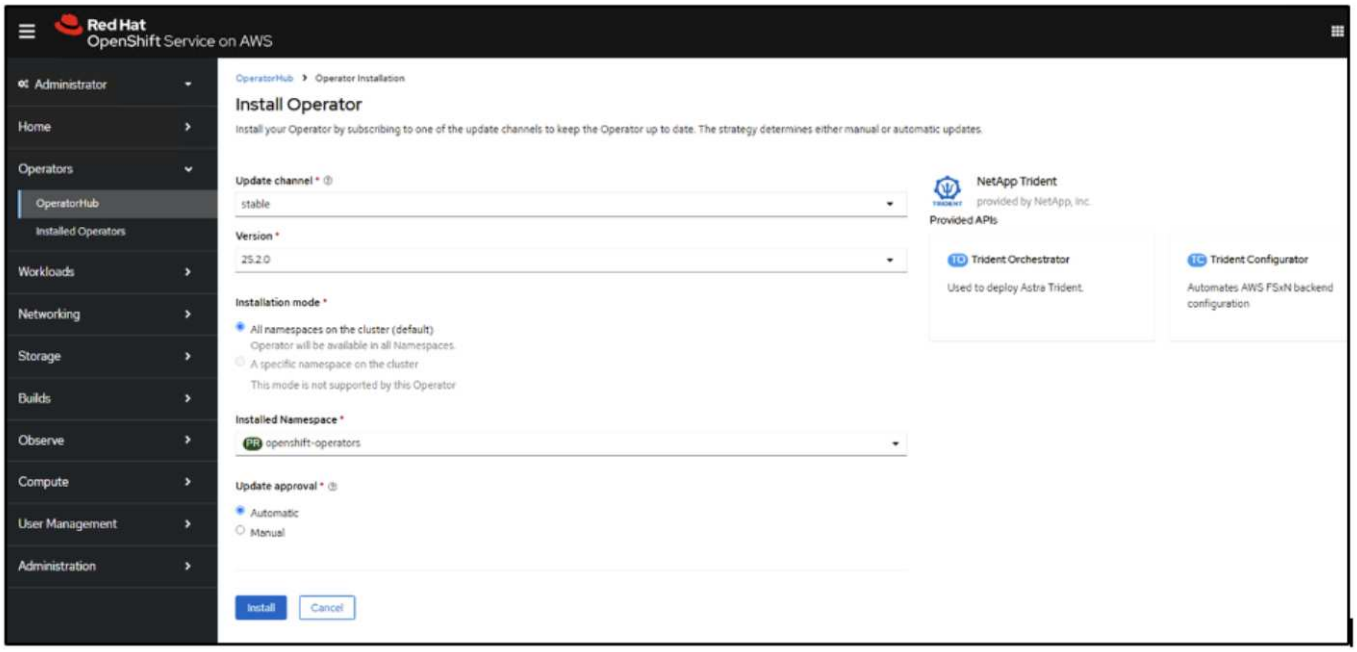
Verwenden Sie das Node-Prep-Flag und installieren Sie Trident wie gezeigt. Bevor Sie den Installationsbefehl ausführen, sollten Sie das Installationspaket heruntergeladen haben. Weitere Informationen finden Sie unter "die Dokumentation hier" .

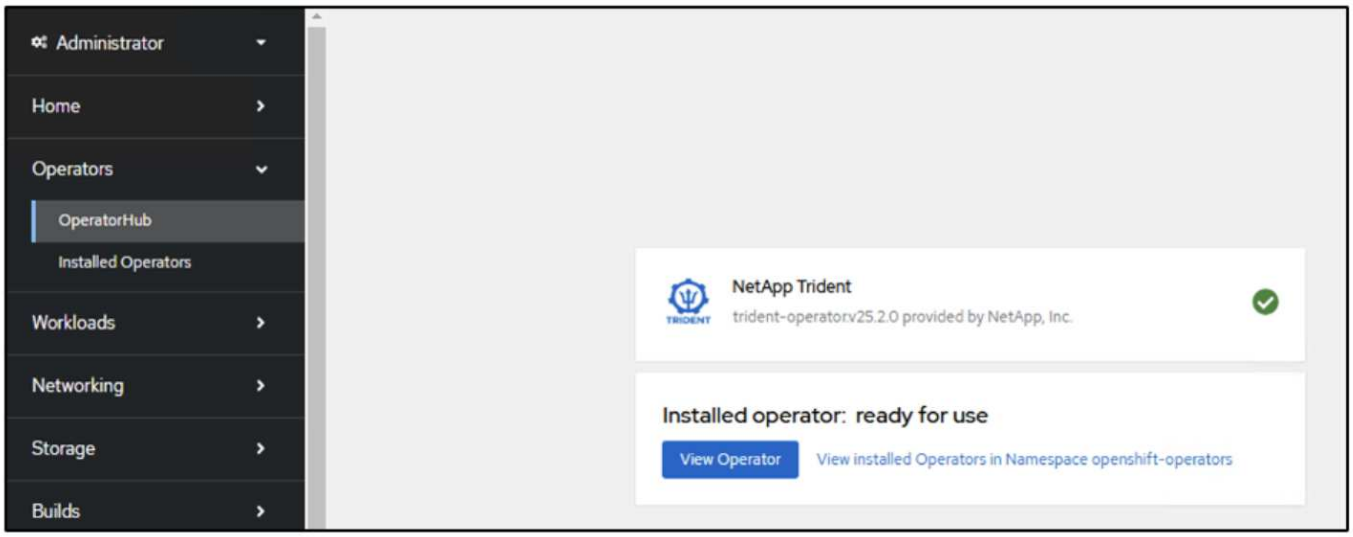
```
#./tridentctl install trident -n trident --node-prep=iscsi
```

Methode 2: Verwenden Sie den Red Hat Certified Trident Operator und passen Sie ihn an. Suchen Sie im OperatorHub den Red Hat Certified Trident Operator und installieren Sie ihn.

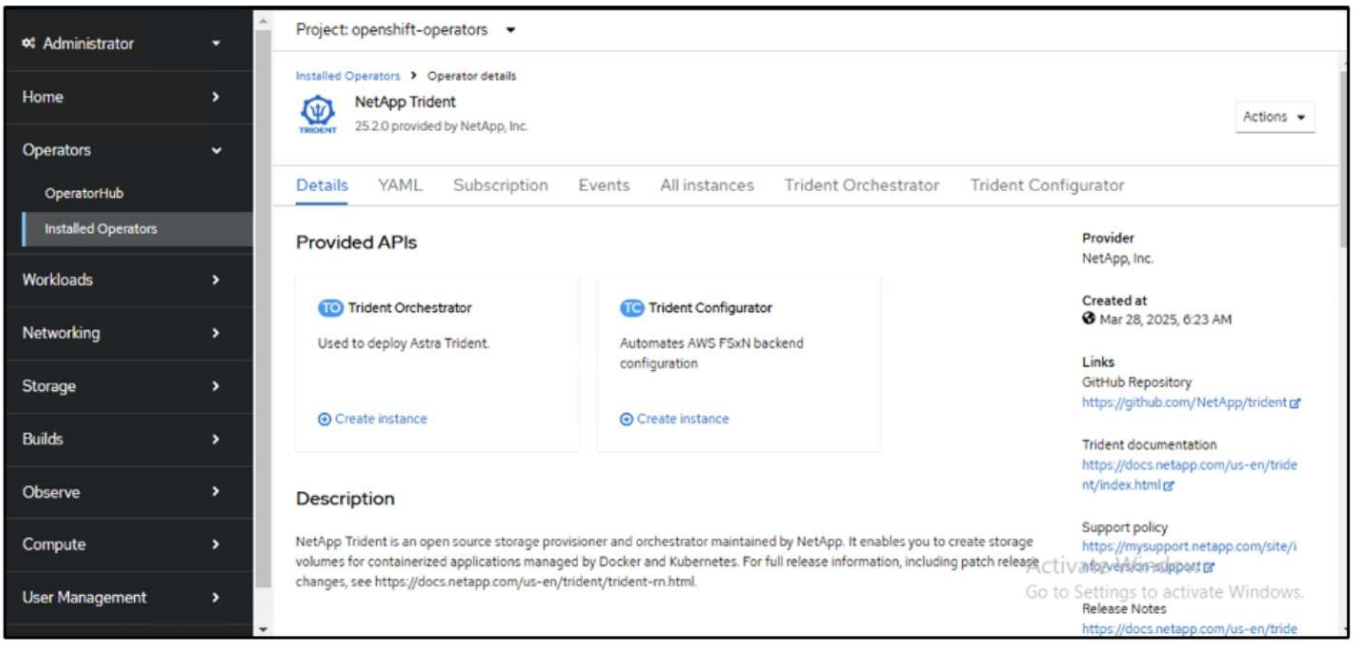


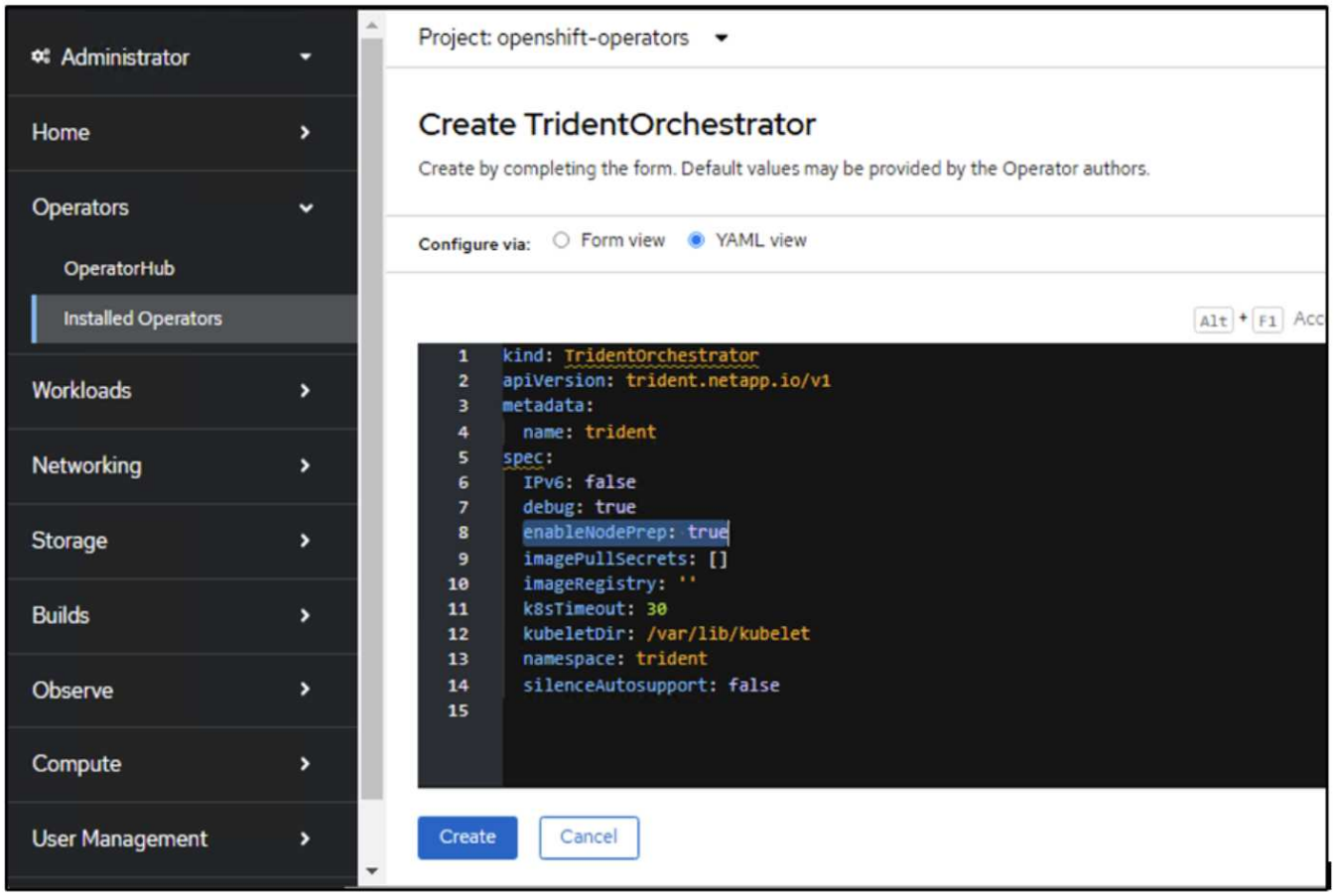

Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
Project: All Projects
OperatorHub
All Items
trident
NetApp Trident
provided by NetApp, Inc.
Trident Operator, to manage NetApp Trident installations
Certified
Community
NetApp Trident
25.2.0 provided by NetApp, Inc.
Install
Channel
stable
Version
25.2.0
Capability level
N/A
Source
Certified
Provider
NetApp, Inc.
Infrastructure features
Container Storage Interface
Disconnected
Repository
Container image
Created at
Mar 9, 2024, 7:00 PM
Support
NetApp
Activate Windows
Go to Settings to activate Windows.

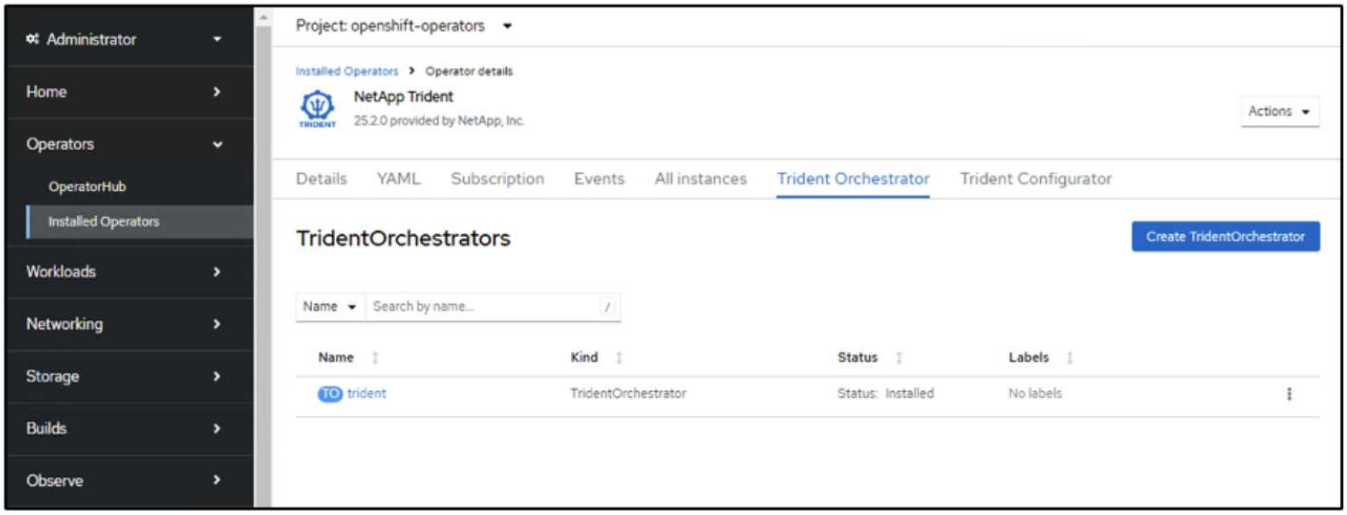

Red Hat
OpenShift Service on AWS
Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
OperatorHub > Operator Installation
Install Operator
Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.
Update channel *
stable
Version *
25.2.0
Installation mode *
All namespaces on the cluster (default)
Operator will be available in all Namespaces.
A specific namespace on the cluster
This mode is not supported by this Operator
Installed Namespace *
openshift-operators
Update approval *
Automatic
Manual
Install
Cancel
NetApp Trident
provided by NetApp, Inc.
Provided APIs
Trident Orchestrator
Used to deploy Astra Trident.
Trident Configurator
Automates AWS FSxN backend configuration



Erstellen Sie als Nächstes die Trident Orchestrator-Instanz. Verwenden Sie die YAML-Ansicht, um benutzerdefinierte Werte festzulegen oder die iscsi-Knotenvorbereitung während der Installation zu aktivieren.








```
[root@localhost RedHat]# oc get pods -n trident
NAME                                  READY   STATUS    RESTARTS   AGE
trident-controller-86f89c855d-8w2jx   6/6     Running   0          38s
trident-node-linux-rnrnn              2/2     Running   0          38s
trident-node-linux-t9bxj              2/2     Running   0          38s
trident-node-linux-vqvl9              2/2     Running   0          38s
[root@localhost RedHat]#
```

Durch die Installation von Trident mit einer der oben genannten Methoden werden die ROSA-Cluster-Worker-Knoten für iSCSI vorbereitet, indem die Dienste iscsid und multipathd gestartet und Folgendes in der Datei /etc/multipath.conf festgelegt wird

```
sh-5.1#
sh-5.1# systemctl status iscsid
● iscsid.service - Open-iSCSI
     Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled; preset: disabled)
     Active: active (running) since Fri 2025-03-21 18:28:13 UTC; 3 days ago
TriggeredBy: ● iscsid.socket
       Docs: man:iscsid(8)
             man:iscsiuio(8)
             man:iscsiadm(8)
   Main PID: 23224 (iscsid)
     Status: "Ready to process requests"
      Tasks: 1 (limit: 1649420)
     Memory: 3.2M
        CPU: 109ms
     CGroup: /system.slice/iscsid.service
             └─23224 /usr/sbin/iscsid -f
sh-5.1#
```

```
sh-5.1#
sh-5.1# systemctl status multipathd
● multipathd.service - Device-Mapper Multipath Device Controller
     Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-03-21 18:20:50 UTC; 3 days ago
TriggeredBy: ● multipathd.socket
   Main PID: 1565 (multipathd)
     Status: "up"
      Tasks: 7
     Memory: 62.4M
        CPU: 33min 51.363s
     CGroup: /system.slice/multipathd.service
             └─1565 /sbin/multipathd -d -s
```

```
sh-5.1#
sh-5.1# cat /etc/multipath.conf
defaults {
    find_multipaths     no
    user_friendly_names yes
}
blacklist {
}
blacklist_exceptions {
    device {
        vendor  NETAPP
        product LUN
    }
}
sh-5.1#
```

c.Überprüfen Sie, ob alle Trident Pods im laufenden Zustand sind.

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-f5f6796f-vd2sk   6/6     Running   0          19h
trident-node-linux-4svgz            2/2     Running   0          19h
trident-node-linux-dj9j4            2/2     Running   0          19h
trident-node-linux-jlshh            2/2     Running   0          19h
trident-node-linux-sqthw            2/2     Running   0          19h
trident-node-linux-ttj9c            2/2     Running   0          19h
trident-node-linux-vmjr5            2/2     Running   0          19h
trident-node-linux-wvqsf            2/2     Running   0          19h
trident-operator-545869857c-kgc7p   1/1     Running   0          19h
[root@localhost hcp-testing]#
```

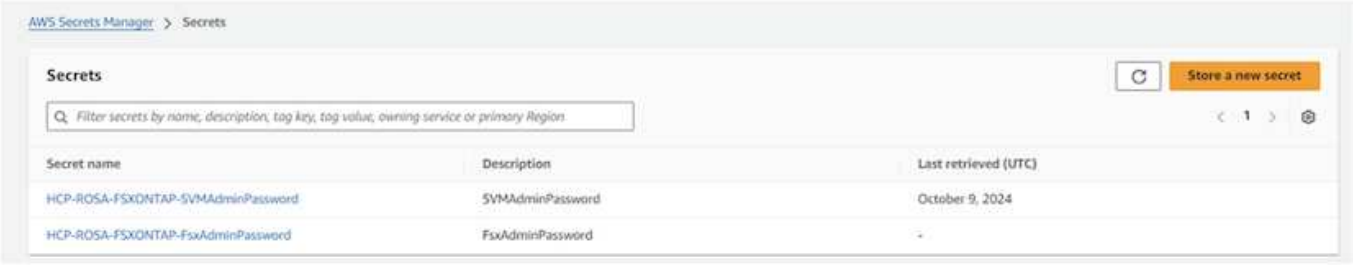
3. Konfigurieren Sie das Trident CSI-Backend für die Verwendung von FSx für ONTAP (ONTAP NAS)

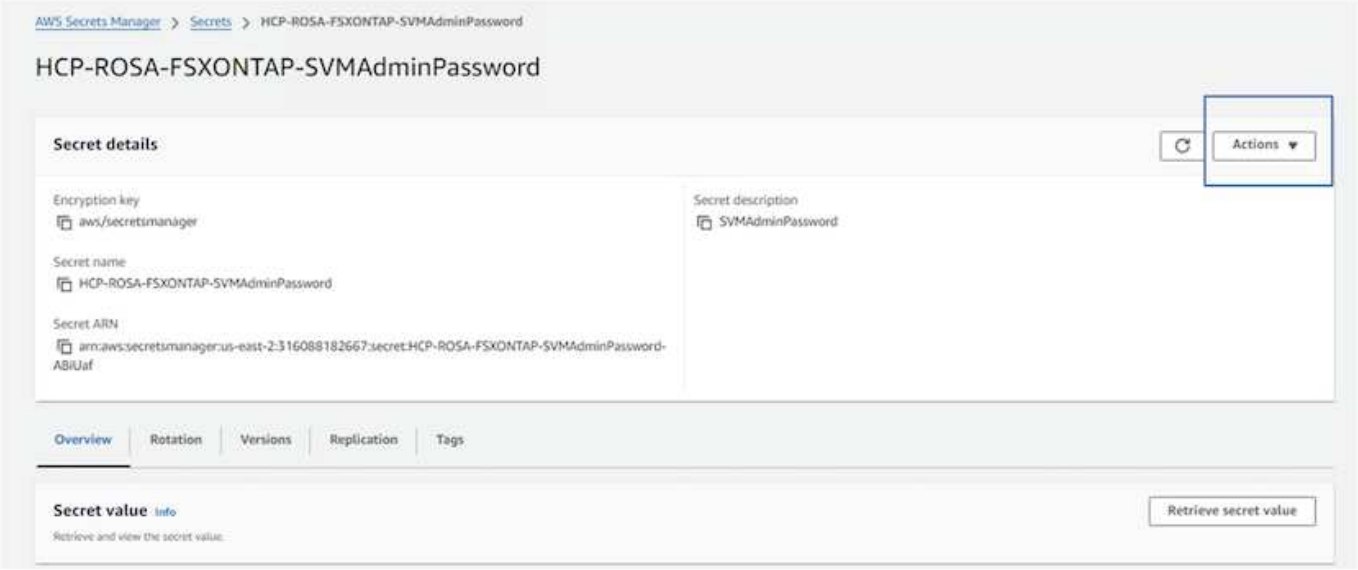
Die Trident Back-End-Konfiguration teilt Trident mit, wie mit dem Speichersystem (in diesem Fall FSx für ONTAP) kommuniziert werden soll. Zum Erstellen des Backends stellen wir die Anmeldeinformationen der Storage Virtual Machine bereit, mit der eine Verbindung hergestellt werden soll, sowie die Clusterverwaltung und die NFS-Datenschnittstellen. Wir verwenden die"ontap-nas-Treiber" um Speichervolumes im FSx-Dateisystem bereitzustellen.

A. Erstellen Sie zunächst ein Geheimnis für die SVM-Anmeldeinformationen mit dem folgenden YAML

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-fsx-ontap-nas-secret
  namespace: trident
type: Opaque
stringData:
  username: vsadmin
  password: <value provided for Define SVM password as a parameter to the
Cloud Formation Stack>
```

Sie können das für FSxN erstellte SVM-Passwort auch wie unten gezeigt vom AWS Secrets Manager abrufen.





b.Fügen Sie als Nächstes mit dem folgenden Befehl das Geheimnis für die SVM-Anmeldeinformationen zum ROSA-Cluster hinzu

```
$ oc apply -f svm_secret.yaml
```

Sie können mit dem folgenden Befehl überprüfen, ob das Geheimnis im Trident-Namespace hinzugefügt wurde

```
$ oc get secrets -n trident |grep backend-fsx-ontap-nas-secret
```

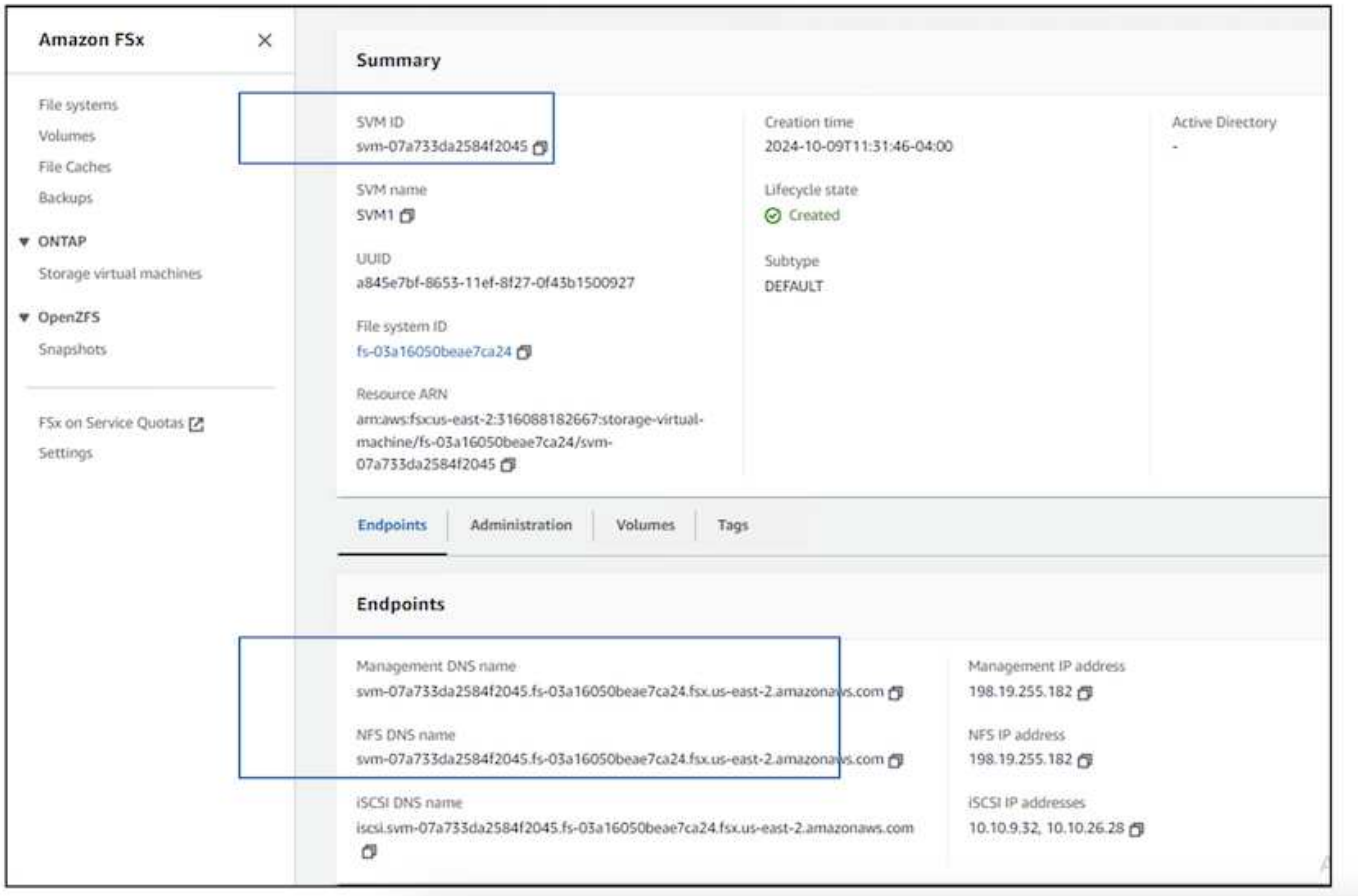
```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc get secrets -n trident | grep backend-fsx-ontap-nas-secret
backend-fsx-ontap-nas-secret         Opaque                  2      21h
[root@localhost hcp-testing]#
```

C. Erstellen Sie als Nächstes das Backend-Objekt. Wechseln Sie dazu in das Verzeichnis **fsx** Ihres geklonten Git-Repositorys. Öffnen Sie die Datei backend-ontap-nas.yaml. Ersetzen Sie Folgendes: **managementLIF** durch den Management-DNS-Namen, **dataLIF** durch den NFS-DNS-Namen der Amazon FSx SVM und **svm** durch den SVM-Namen. Erstellen Sie das Backend-Objekt mit dem folgenden Befehl.

Erstellen Sie das Backend-Objekt mit dem folgenden Befehl.

```
$ oc apply -f backend-ontap-nas.yaml
```

Sie können den Management-DNS-Namen, den NFS-DNS-Namen und den SVM-Namen von der Amazon FSx Konsole abrufen, wie im folgenden Screenshot gezeigt



D. Führen Sie nun den folgenden Befehl aus, um zu überprüfen, ob das Backend-Objekt erstellt wurde und die Phase „Gebunden“ und der Status „Erfolgreich“ anzeigt.

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc apply -f backend-ontap-nas.yaml
tridentbackendconfig.trident.netapp.io/backend-fsx-ontap-nas created
[root@localhost hcp-testing]# oc get tbc -n trident
NAME                  BACKEND NAME   BACKEND UUID                           PHASE   STATUS
backend-fsx-ontap-nas   fsx-ontap      acc65405-56be-4719-999d-27b448a50e29   Bound   Success
[root@localhost hcp-testing]#
```

4. Speicherklasse erstellen Nachdem das Trident Backend konfiguriert ist, können Sie eine Kubernetes-Speicherklasse erstellen, um das Backend zu verwenden. Die Speicherklasse ist ein Ressourcenobjekt, das dem Cluster zur Verfügung gestellt wird. Es beschreibt und klassifiziert den Speichertyp, den Sie für eine Anwendung anfordern können.

A. Überprüfen Sie die Datei storage-class-csi-nas.yaml im Ordner fsx.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: trident-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: True
reclaimPolicy: Retain
```

B. Erstellen Sie eine Speicherklasse im ROSA-Cluster und überprüfen Sie, ob die Speicherklasse Trident-CSI erstellt wurde.

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc apply -f storage-class-csi-nas.yaml
storageclass.storage.k8s.io/trident-csi created
[root@localhost hcp-testing]# oc get sc
NAME               PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
gp2-csi            ebs.csi.aws.com         Delete          WaitForFirstConsumer   true                   2d16h
gp3-csi (default)  ebs.csi.aws.com         Delete          WaitForFirstConsumer   true                   2d16h
trident-csi        csi.trident.netapp.io   Retain          Immediate              true                   4s
[root@localhost hcp-testing]#
```

Damit ist die Installation des Trident CSI-Treibers und seine Verbindung zum FSx für das ONTAP Dateisystem abgeschlossen. Jetzt können Sie eine Beispielanwendung mit PostgreSQL-Status auf ROSA mithilfe von Dateivolumes auf FSx für ONTAP bereitstellen.

C. Stellen Sie sicher, dass keine PVCs und PVs mit der Speicherklasse Trident-CSI erstellt wurden.

D. Überprüfen Sie, ob Anwendungen mit Trident CSI PV erstellen können.

Erstellen Sie ein PVC mit der Datei pvc-trident.yaml im Ordner **fsx**.

```
pvc-trident.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: trident-csi
```

```
 You can issue the following commands to create a pvc and verify that it
has been created.
image:redhat-openshift-container-rosa-011.png["Erstellen Sie Test-PVC mit
Trident"]
```

Um iSCSI zu verwenden, sollten Sie iSCSI wie zuvor gezeigt auf den Worker-Knoten aktiviert haben und ein iSCSI-Backend und eine Speicherklasse erstellen. Hier sind einige Beispieldateien im YAML-Format.

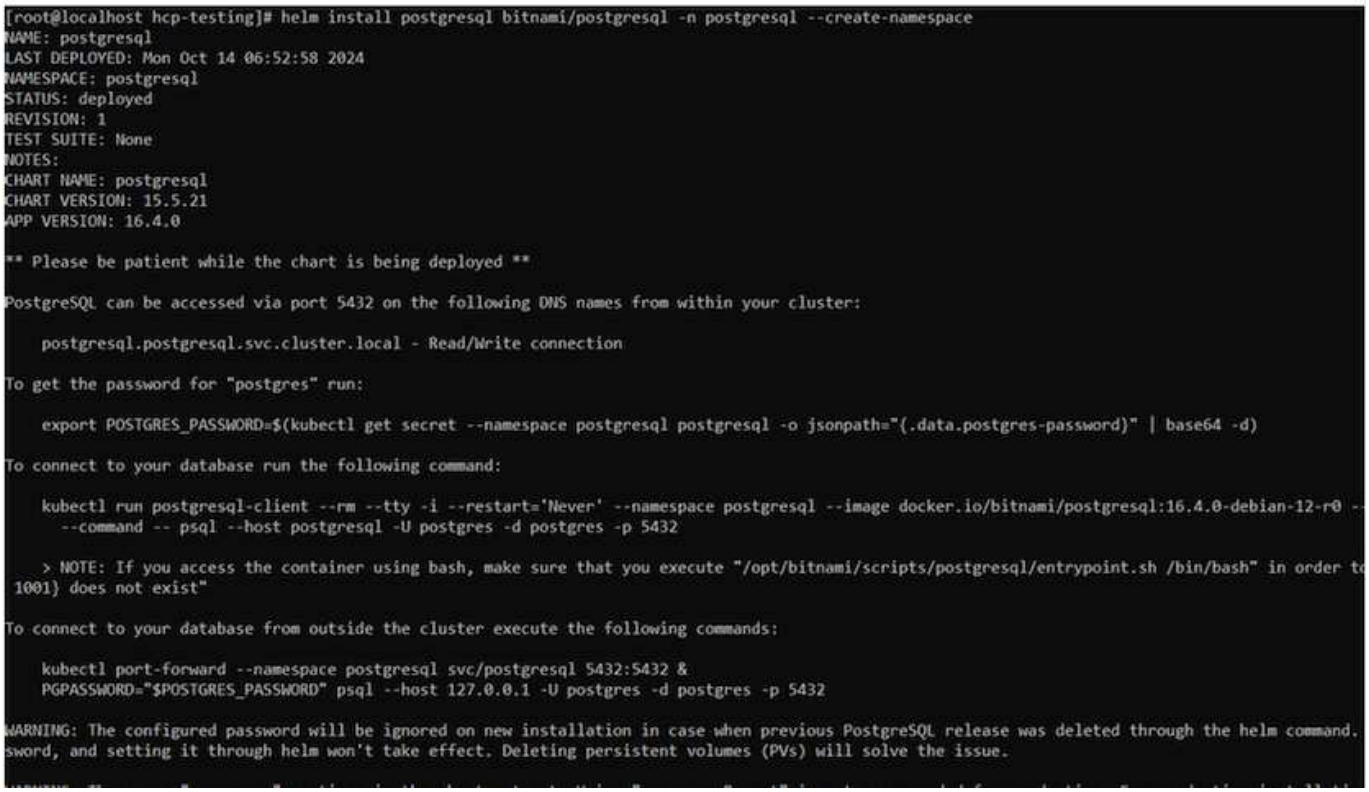
```
cat tbc.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: fsxadmin
  password: <password for the fsxN filesystem>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  storageDriverName: ontap-san
  managementLIF: <management lif of fsxN filesystem>
  backendName: backend-tbc-ontap-san
  svm: svm_FSxNForROSAiSCSI
  credentials:
    name: backend-tbc-ontap-san-secret

cat sc.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: trident-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
allowVolumeExpansion: true
```

5. Stellen Sie eine Beispielanwendung mit PostgreSQL-Status bereit

A. Verwenden Sie Helm, um PostgreSQL zu installieren

```
$ helm install postgresql bitnami/postgresql -n postgresql --create
-namespace
```



B. Stellen Sie sicher, dass der Anwendungs-Pod ausgeführt wird und ein PVC und PV für die Anwendung erstellt wurde.

C. Stellen Sie einen Postgresql-Client bereit

Verwenden Sie den folgenden Befehl, um das Kennwort für den installierten PostgreSQL-Server abzurufen.

```
$ export POSTGRES_PASSWORD=$(kubectl get secret --namespace postgresql
postgresql -o jsoata.postgres-password}" | base64 -d)
```

Verwenden Sie den folgenden Befehl, um einen PostgreSQL-Client auszuführen und mit dem Kennwort eine Verbindung zum Server herzustellen

```
$ kubectl run postgresql-client --rm --tty -i --restart='Never'
--namespace postgresql --image docker.io/bitnami/postgresql:16.2.0-debian-
11-r1 --env="PGPASSWORD=$POSTGRES_PASSWORD" \
> --command -- psql --host postgresql -U postgres -d postgres -p 5432
```

```
[root@localhost hcp-testing]# kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitna
$POSTGRES_PASSWORD" \
> --command -- psql --host postgresql -U postgres -d postgres -p 5432
Warning: would violate PodSecurity "restricted:v1.24": allowPrivilegeEscalation != false (container "postgresql-client" must set securityC
 capabilities (container "postgresql-client" must set securityContext.capabilities.drop=["ALL"]), runAsNonRoot != true (pod or container "
Root=true), seccompProfile (pod or container "postgresql-client" must set securityContext.seccompProfile.type to "RuntimeDefault" or "Loca
If you don't see a command prompt, try pressing enter.
```

D. Erstellen Sie eine Datenbank und eine Tabelle. Erstellen Sie ein Schema für die Tabelle und fügen Sie 2 Datenzeilen in die Tabelle ein.

```
postgres=# CREATE DATABASE erp;
CREATE DATABASE
postgres=# \c erp
psql (16.2, server 16.4)
You are now connected to database "erp" as user "postgres".
erp=# CREATE TABLE PERSONS(ID INT PRIMARY KEY NOT NULL, FIRSTNAME TEXT NOT NULL, LASTNAME TEXT NOT NULL);
CREATE TABLE
erp=# INSERT INTO PERSONS VALUES(1,'John','Doe');
INSERT 0 1
erp=# \dt
         List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | persons | table | postgres
(1 row)
```

```
erp=# SELECT * FROM PERSONS;
 id | firstname | lastname
----+-----------+----------
  1 | John      | Doe
(1 row)
```

```
erp=# INSERT INTO PERSONS VALUES(2,'Jane','Scott');
INSERT 0 1
erp=# SELECT * from PERSONS;
 id | firstname | lastname
----+-----------+----------
  1 | John      | Doe
  2 | Jane      | Scott
(2 rows)
```

Red Hat OpenShift Service auf AWS mit NetApp ONTAP

In diesem Dokument wird beschrieben, wie Sie NetApp ONTAP mit dem Red Hat OpenShift Service auf AWS (ROSA) verwenden.

Volume-Snapshot erstellen

1. Erstellen Sie einen Snapshot des App-Volumes. In diesem Abschnitt zeigen wir, wie Sie einen Trident-Snapshot des mit der App verknüpften Volumes erstellen. Dies ist eine zeitpunktbezogene Kopie der App-Daten. Wenn die Anwendungsdaten verloren gehen, können wir die Daten aus dieser Point-in-Time-Kopie wiederherstellen. HINWEIS: Dieser Snapshot wird im selben Aggregat wie das Originalvolume in ONTAP(vor Ort oder in der Cloud) gespeichert. Wenn also das ONTAP Speicheraggregat verloren geht, können wir die App-Daten nicht aus seinem Snapshot wiederherstellen.

**A. Erstellen Sie eine VolumeSnapshotClass. Speichern Sie das folgende Manifest in einer Datei namens volume-snapshot-class.yaml

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
 name: fsx-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Erstellen Sie mithilfe des obigen Manifests einen Snapshot.

```
[root@localhost hcp-testing]# oc create -f volume-snapshot-class.yaml
volumesnapshotclass.snapshot.storage.k8s.io/fsx-snapclass created
[root@localhost hcp-testing]#
```

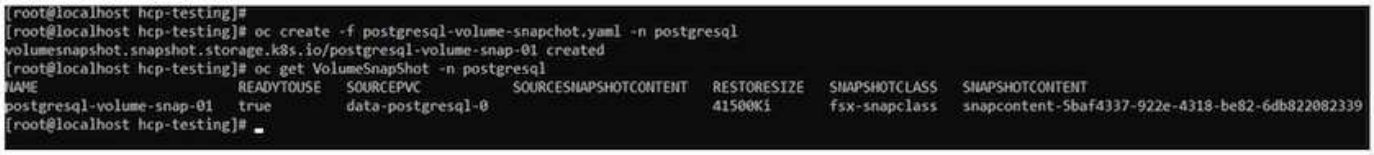
B. Erstellen Sie als Nächstes einen Snapshot. Erstellen Sie einen Snapshot des vorhandenen PVC, indem Sie VolumeSnapshot erstellen, um eine zeitpunktbezogene Kopie Ihrer Postgresql-Daten zu erstellen. Dadurch wird ein FSx-Snapshot erstellt, der im Dateisystem-Backend fast keinen Platz einnimmt. Speichern Sie das

folgende Manifest in einer Datei namens volume-snapshot.yaml:

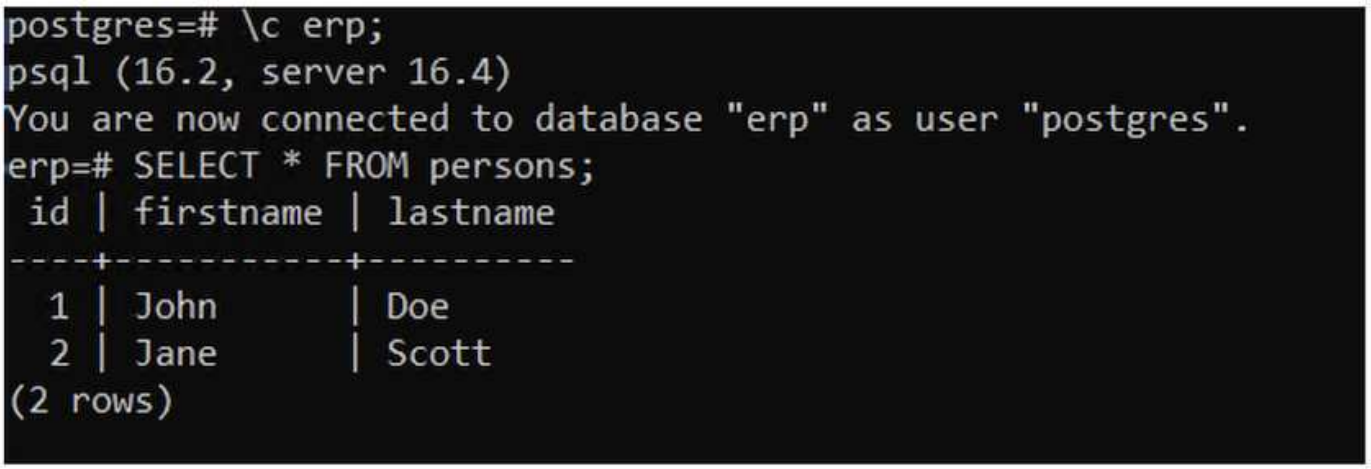
```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
 name: postgresql-volume-snap-01
spec:
 volumeSnapshotClassName: fsx-snapclass
 source:
   persistentVolumeClaimName: data-postgresql-0
```

C. Erstellen Sie den Volume-Snapshot und bestätigen Sie, dass er erstellt wurde

Löschen Sie die Datenbank, um den Datenverlust zu simulieren (Datenverlust kann aus verschiedenen Gründen auftreten, hier simulieren wir ihn nur durch Löschen der Datenbank).



D. Löschen Sie die Datenbank, um den Datenverlust zu simulieren (Datenverlust kann aus verschiedenen Gründen auftreten, hier simulieren wir ihn nur durch Löschen der Datenbank)



Wiederherstellen aus Volume-Snapshot

1. Wiederherstellen aus Snapshot In diesem Abschnitt zeigen wir, wie Sie eine Anwendung aus dem Trident-Snapshot des App-Volumes wiederherstellen.

A. Erstellen Sie einen Volume-Klon aus dem Snapshot

Um den vorherigen Zustand des Volumes wiederherzustellen, müssen Sie basierend auf den Daten im von Ihnen erstellten Snapshot einen neuen PVC erstellen. Speichern Sie dazu das folgende Manifest in einer Datei mit dem Namen pvc-clone.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: postgresql-volume-clone
spec:
 accessModes:
   - ReadWriteOnce
 storageClassName: trident-csi
 resources:
   requests:
     storage: 8Gi
 dataSource:
   name: postgresql-volume-snap-01
   kind: VolumeSnapshot
   apiGroup: snapshot.storage.k8s.io
```

Erstellen Sie einen Klon des Volumes, indem Sie mithilfe des obigen Manifests einen PVC erstellen und dabei den Snapshot als Quelle verwenden. Wenden Sie das Manifest an und stellen Sie sicher, dass der Klon erstellt wird.

```
[root@localhost hcp-testing]# oc create -f postgresql-pvc-clone.yaml -n postgresql
persistentvolumeclaim/postgresql-volume-clone created
[root@localhost hcp-testing]# oc get pvc -n postgresql
NAME                     STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
data-postgresql-0        Bound    pvc-e3ddd9bd-e6a7-4a4a-b935-f1c090fd8db6   8Gi        RWO            trident-csi
postgresql-volume-clone  Bound    pvc-b38fbc54-55dc-47e8-934d-47f181fddac6   8Gi        RWO            trident-csi
[root@localhost hcp-testing]#
```

B. Löschen Sie die ursprüngliche PostgreSQL-Installation

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# helm uninstall postgresql -n postgresql
release "postgresql" uninstalled
[root@localhost hcp-testing]# oc get pods -n postgresql
No resources found in postgresql namespace.
[root@localhost hcp-testing]#
```

C. Erstellen Sie eine neue PostgreSQL-Anwendung mit dem neuen Klon-PVC

```
$ helm install postgresql bitnami/postgresql --set
primary.persistence.enabled=true --set
primary.persistence.existingClaim=postgresql-volume-clone -n postgresql
```

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# helm install postgresql bitnami/postgresql --set primary.persistence.enabled=true \
> --set primary.persistence.existingClaim=postgresql-volume-clone -n postgresql
NAME: postgresql
LAST DEPLOYED: Mon Oct 14 12:03:31 2024
NAMESPACE: postgresql
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: postgresql
CHART VERSION: 15.5.21
APP VERSION: 16.4.0

** Please be patient while the chart is being deployed **

PostgreSQL can be accessed via port 5432 on the following DNS names from within your cluster:

    postgresql.postgresql.svc.cluster.local - Read/Write connection

To get the password for "postgres" run:

    export POSTGRES_PASSWORD=$(kubectl get secret --namespace postgresql postgresql -o jsonpath="{.data.postgres-password}" | ba

To connect to your database run the following command:

    kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitnami/postgresql:16
      --command -- psql --host postgresql -U postgres -d postgres -p 5432

    > NOTE: If you access the container using bash, make sure that you execute "/opt/bitnami/scripts/postgresql/entrypoint.sh /b
 1001} does not exist"

To connect to your database from outside the cluster execute the following commands:

    kubectl port-forward --namespace postgresql svc/postgresql 5432:5432 &
    PGPASSWORD="$POSTGRES_PASSWORD" psql --host 127.0.0.1 -U postgres -d postgres -p 5432

WARNING: The configured password will be ignored on new installation in case when previous PostgreSQL release was deleted throug
sword, and setting it through helm won't take effect. Deleting persistent volumes (PVs) will solve the issue.

WARNING: There are "resources" sections in the chart not set. Using "resourcesPreset" is not recommended for production. For pro
ng to your workload needs:
  - primary.resources
  - readReplicas.resources
+info https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
[root@localhost hcp-testing]#
```

D. Überprüfen Sie, ob sich der Anwendungs-Pod im Ausführungsstatus befindet

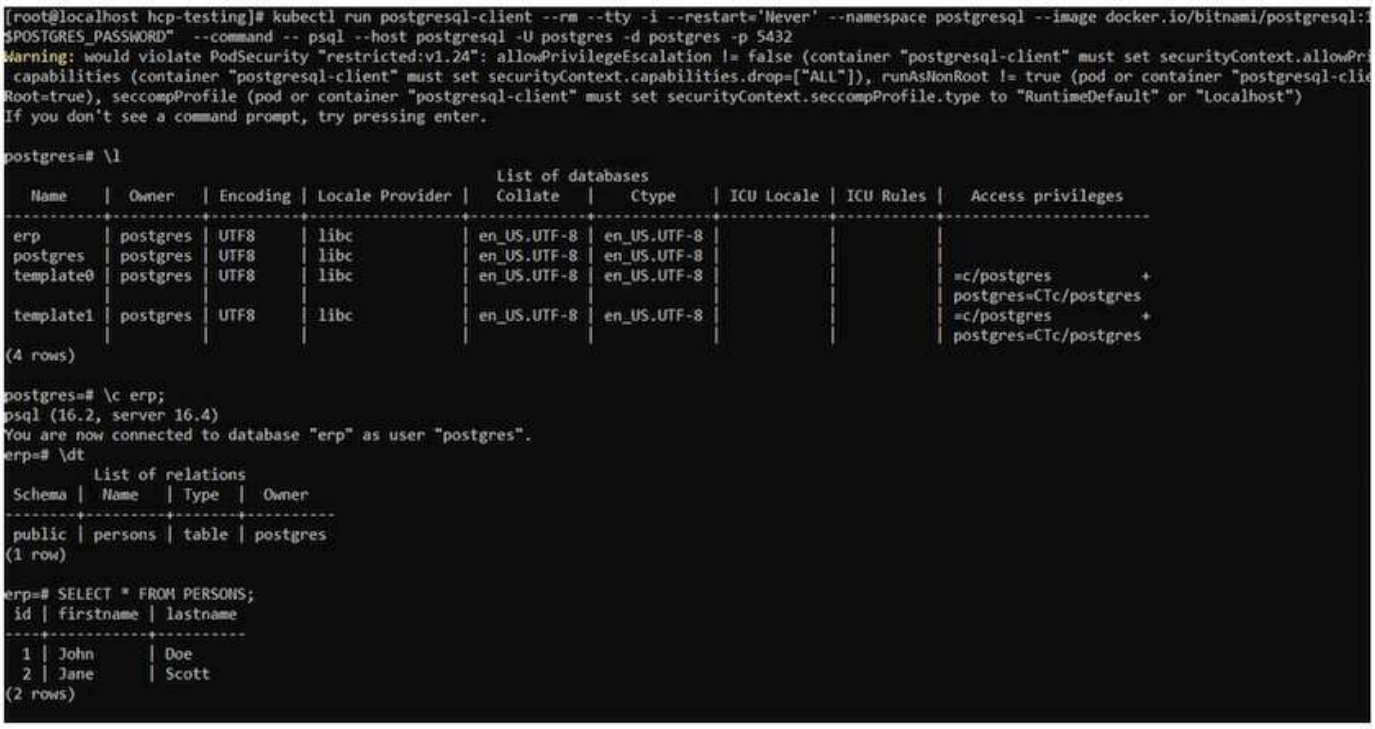
```
[root@localhost hcp-testing]# oc get pods -n postgresql
NAME           READY   STATUS    RESTARTS   AGE
postgresql-0   1/1     Running   0          2m1s
[root@localhost hcp-testing]#
```

e. Überprüfen Sie, ob der Pod den Klon als PVC verwendet

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc describe pod/postgresql-0 -n postgresql
```

```
  ContainersReady             True
  PodScheduled                True
Volumes:
  empty-dir:
    Type:       EmptyDir (a temporary directory that shares a pod's lifetime)
    Medium:
    SizeLimit:  <unset>
  dshm:
    Type:       EmptyDir (a temporary directory that shares a pod's lifetime)
    Medium:     Memory
    SizeLimit:  <unset>
  data:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:  postgresql-volume-clone
    ReadOnly:   false
QoS Class:      Burstable
Node-Selectors: <none>
Tolerations:    node.kubernetes.io/memory-pressure:NoSchedule op=Exists
                node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason                  Age    From                     Message
  ----    ------                  ----   ----                     -------
  Normal  Scheduled               3m55s  default-scheduler        Successfully assigned postgresql/postgres
.us-east-2.compute.internal
  Normal  SuccessfulAttachVolume  3m54s  attachdetach-controller  AttachVolume.Attach succeeded for volume
8-934d-47f181fddac6"
  Normal  AddedInterface          3m43s  multus                   Add eth0 [10.129.2.126/23] from ovn-kuber
  Normal  Pulled                  3m43s  kubelet                  Container image "docker.io/bitnami/postgr
r0" already present on machine
  Normal  Created                 3m42s  kubelet                  Created container postgresql
  Normal  Started                 3m42s  kubelet                  Started container postgresql
[root@localhost hcp-testing]#
```

f) Um zu überprüfen, ob die Datenbank wie erwartet wiederhergestellt wurde, gehen Sie zurück zur Containerkonsole und zeigen Sie die vorhandenen Datenbanken an



Demo-Video

Amazon FSx for NetApp ONTAP mit Red Hat OpenShift Service auf AWS unter Verwendung der gehosteten Steuerebene

Weitere Videos zu Red Hat OpenShift und OpenShift-Lösungen finden Sie"hier," .